Amendments to the Claims:

This listing of claims replaces all prior versions and listings of claims in the application:

Listing of Claims:

1. (Original) A method of debugging code that executes in a multithreaded processor having a plurality of microengines comprises:

receiving a program instruction and an identification representing a selected one of the plurality of microengines from a remote user interface connected to the processor;

pausing program execution in the threads executing in the selected microengine;

inserting a breakpoint after a program instruction in the selected microengine that matches the program instruction received from the remote user interface;

resuming program execution in the selected microengine;

executing a breakpoint routine if program execution in the selected microengine encounters the breakpoint; and

resuming program execution in the microengine.

2. (Original) The method of claim 1 wherein pausing comprises disabling a processor enable bit associated with the selected microengine.

3. (Original) The method of claim 1 wherein pausing comprises:

determining when a context swap between the threads occurs in the selected microengine; and

disabling a processor enable bit associated with the selected microengine in response to the context swap.

4. (Original) The method of claim 1 wherein executing comprises:

sending an interrupt to a controlling processor register; and
processing the interrupt.

5. (Original) The method of claim 4 wherein processing comprises:
sending the identification to an interrupt handler; and
executing the breakpoint routine in the microengine represented by the identification.

6. (Original) The method of claim 1 wherein the breakpoint routine comprises:
writing data to a register.

7. (Original) The method of claim 6 wherein the data are representative of the state of the threads in the selected microengine.

8. (Original) The method of claim 4 wherein the controlling processor register comprises a 32-bit register.

9. (Original) The method of claim 8 wherein bits 6:0 of the 32-bit register represent thread numbers corresponding to the threads in the selected microengine.

10. (Original) The method of claim 8 where bits 9:7 of the 32-bit register represent whether the interrupt is a breakpoint interrupt.

11. (Original) The method of claim 1 wherein the breakpoint routine resides in a store of a controlling processor.

12. (Currently amended) A processor that can execute multiple contexts and that comprises:
a register stack;

a program counter for each executing context;

an arithmetic logic unit coupled to the register stack and a program control store that stores a breakpoint command that causes the processor to:

perform a breakpoint routine residing in a debug library in one of the contexts, a breakpoint in the context pointing to the breakpoint routine, the breakpoint inserted into the context in response to a user request received through a remote user interface connected to the processor, the request including a context identification; and

resume program execution.

Claims 13-14. (Canceled)

15. (Currently amended) The processor of claim 12 ~~13~~ wherein an end of the breakpoint routine points to a program counter of the context.

16. (Original) The processor of claim 12 wherein the breakpoint routine performs at a context switch.

17. (Currently amended) The processor of claim 12 ~~13~~ wherein the breakpoint causes an interrupt.

18. (Original) The processor of claim 17 wherein an interrupt handler services the interrupt.

19. (Original) The processor of claim 18 wherein the interrupt handler identifies the context from the interrupt.

20. (Original) The processor of claim 18 wherein the interrupt handler identifies a processor identification.

21. (Original) A computer program product, disposed on a computer readable medium, the product including instructions for causing a multithreaded processor having a plurality of microengines to:

receive a program instruction and an identification representing a selected one of the plurality of microengines from a remote user interface connected to the processor;

pause program execution in the threads executing in the selected microengine;

insert a breakpoint after a program instruction in the selected microengine that matches the program instruction received from the remote user interface;

resume program execution in the selected microengine;

execute a breakpoint routine if program execution in the selected microengine encounters the breakpoint; and

resume program execution in the microengine.